ULTRACOMPUTER NOTE 63

THE ULTRACOMPUTER AS A VEHICLE FOR POLYMER SIMULATIONS

by

MARVIN BISHOP

OCTOBER 1983

INTRODUCTION

Computer science is intimately linked to microelectronics[1]. Hardware technology has advanced significantly since the construction of the first computers. Today, very large scale integration (VLSI) circuits on single silicon chips have replaced the myriad of tubes and wires of early machines. The density of logical devices on chips has increased while production costs have fallen. However, there is an ultimate limitation on the operating speed of the computer components--a signal cannot be propagated through an electrical conductor faster than the speed of light. But speedups of serial processing can be achieved by performing more than one operation simultaneously. "The future of high performance computers is in parallel computation".[2]

In this note key concepts of parallelism relating to both hardware and software are reviewed. The NYU Ultracomputer design is treated. One particular application area--- the simulation of polymer systems via the technique of molecular dynamics---is discussed at length.

PARALLELISM IN COMPUTER HARDWARE AND SOFTWARE

Methods for achieving parallel operation depend upon replicating the instruction stream and/or data stream. Thus, there are four classes of computers[3] : single - instruction single - data stream (SISD), single - instruction multiple - data stream (SIMD), multiple - instruction single - data stream (MISD), and multiple - instruction multiple - data stream (MIMD). The SISD machine (e.g. IBM 360) is the canonical serial computer---individual steps are performed on individual items of data one after another in time sequence. A SIMD machine (e.g. CRAY-1) can use pipelines to operate on vectors of data. In pipelining[4] a sequential computational procedure is broken down into stages and separate hardware units are provided for carrying out the computation of each stage. A pipeline is analogous to an assembly line in that all operations are conducted simultaneously but not on the same material.

Alternatively[5], an SIMD computer (e.g. ILLIAC IV) can be set up as an array of processors. A single master control unit sends instructions to independent processing elements, PEs, each with its own memory. Each PE executes exactly the same instruction, but on different data. By a mask, it is possible to cause any PE to idle rather than actually execute an instruction.

The central questions for SIMD computers relate to the ability to cast a computation into vector form. In contrast, an MIMD machine (e.g. C.mmp) has difficulties in making independent computers cooperate so that one problem can be partitioned among the PEs[6]. It is interesting to note that many of the desirable features of a MIMD machine were indicated as early as 1966[7]: core storage uniformly addressable, collective instructions which eliminate a master PE, logical distinction between various PEs and parallel compilers. These ideas have been extended to define an ideal parallel computer ("paracomputer"[8]). This machine is composed of identical PEs sharing a common memory. The PEs may simultaneously read or write any memory cell in one cycle provided the "serialization principle" is followed. When simultaneous requests are issued to the same memory cell by many PEs, the effect is as though the requests occurred in some (unspecified) serial order.

A simulator ("WASHCLOTH"[9]) has been developed to evaluate the paracomputer design. A fetch-and-add instruction has been introduced in order to handle the communication between PEs; e.g. critical sections controlled by a semaphore can be programmed in terms of the fetch-and-add instruction. Its format is F&A(V,e), where V is an integer variable and e is an integer expression. This operation returns V and replaces the contents of storage location V by the sum V + e. If many fetch-and-add operations simultaneously address V, then there exists some serial execution order of processors 1 and 2 say, such that each operation yields the value corresponding to its position in this order and such that V receives the appropriate total increment.

For example if V initially contained the value 10 and processors 1 and 2 execute F&A(V,2) and F&A(V,6) respectively, then after the simultaneous executions V will contain 18 and either S_1=10 and S_2=12 or S_1=16 and S_2=10 depending upon whether or not processor 1 executed fetch-and-add before processor 2. S_1 and S_2 are the results of the fetch-and-add on processor 1 and 2 respectively.[10]

This software simulator is being used to study applications of the NYU Ultracomputer[11,12,13]. The Ultracomputer is a proposed MIMD parallel machine composed of 4096 autonomous PEs. It will use an enhanced message switching network with the geometry of an Omega-network. The plan is to have local memory implemented as a cache to minimize the effect of network latency and to reduce network traffic. The switches will have some memory and some elementary processing capabilities so that the bandwidth (maximum total throughput of the network) will be proportional to the number of PEs. These hardware issues are being investigated by profiling various codes.

The simulator has been used to examine a number of scientific computations[14] including Monte Carlo calculations, Navier-Stokes equations, reduction of symmetric matrices to tridiagonal form and atmospheric modelling. Efficiencies were obtained for a variety of input size/number of PE combinations via the following schema. If T_p is the number of unit time steps required to perform a calculation using p processors, the speedup of the p-processor calculation over a uniprocessor (requiring time T_1) is $S_p = T_1/T_p$ and the efficiency of the calculation is $E_p = S_p/p$. This is the actual speedup divided by

the maximum possible speedup using p processors. It was concluded that "very large problems can be efficiently dealt with by large processor arrays but that small problems are best left to one or a few processor units."[14]

POLYMER SIMULATIONS

The development of polymer science has had a profound effect on technology. Polymeric materials have important applications because of their mechanical, dielectric, flow and thermal properties. These physical properties are determined by the molecular configurations of the polymer. For example, a polymer in solution which is tightly coiled will have a lower viscosity than when it is extended because the tightly coiled structure presents a smaller cross sectional area to the solvent molecules. Clearly, a fundamental area of polymer research is the investigation of the determining factors of molecular configurations and the mechanisms by which the polymer chain moves from one configuration to another.

A number of theoretical models have been developed. In early models[15] a polymer chain was represented by a random walk between points in space. The points represent the polymer atoms and the steps the interatomic bonds. In a "pure" random walk intersections are allowed. The simplification of Gaussian statistics was used to calculate analytically many properties. However, the repulsive forces between atoms prevent such intersections in real polymers. The addition of excluded volume effects to the random walk makes the model too complicated for exact calculation. Computer Monte Carlo[16] methods have been employed to examine model polymer systems. Many models assume a lattice and forbid more than one chain segment from occupying

the same site. Lattice models gain computational simplicity at the expense of making space discrete.

A few studies have also been done for continuous polymers and for more realistic potentials[17]. All of these simulations have employed the standard Metropolis Monte Carlo[18] procedure and, as such, have provided no information about chain dynamics. Time dependent data are crucial for understanding the transport and dielectric behavior of polymers.

An alternative to the Monte Carlo method exists. The molecular dynamics[19] (MD) technique simulates motion by a numerical solution of Newton's equations. An interaction potential is selected to represent the interactions between particles of the system. This potential is usually chosen to be pairwise additive and of finite range, R_c, such that for distances greater than R_c, the potential is zero. The equations of motion describe the evolution of the system in terms of the forces(i.e.derivatives of the potential function). These equations are coupled, second-order differential equations and can be integrated via a finite difference scheme such as the one proposed by Verlet[20]. The computer solution generates the sequence of states of the system at discrete time intervals. Suitable time averaging yields information about both static and dynamic properties.

The calculational bottleneck is the computation of the forces. There are $n(n-1)/2$ pair interactions in a system with n particles. To compute all of these is a $O(n^2)$ process. However, because of the

finite range of the potential, most of the terms do not contribute to the total force. Box scheme methods[21] have been introduced to improve the efficiency to O(n). The system is divided into equal sized boxes and a tabulation of the particles in each box is made. Then the full force sum is replaced by a sum over particles in each box and a sum over particles in neighboring boxes. As particles move the contents of the boxes change and the particle tabulation must be updated. A parallel implementation of this method has been described[22] in the somewhat different context of Monte Carlo calculations. Neighbors of particles were examined in parallel rather than serially. It was found that parallelization "will be very effective for carrying out very large Monte Carlo calculations."[22] MD parallelizations should be even more effective since the Monte Carlo algorithm has some inherent serial aspects whereas the integration of Newton's equations is natural to carry out in parallel.

MD has already been applied to the study of liquid n-butane by Ryckaert and Bellemans[23] and by Weber[24]. Solvent effects for short chains have been probed by Bishop, Kalos and Frisch;[25] by Rapaport;[26] by Rebertus, Berne and Chandler[27] and by Bruns and Bansal[28].

However, it is only with long polymers that certain properties appear. For example, it has been experimentally determined that the viscosity of almost all polymers varies as a power of polymer length and that the power changes from a value of 1 for small polymers to a value of 3.4 for long polymers[29]. The change takes place at a critical length of about 1000 monomer units. The "particles" (beads) in our

polymer model represent statistical segments of the polymer chain and, hence, the number of beads per chain necessary to observe this transition will be much less than 1000. Nevertheless, the computer time needed to undertake a serial calculation is huge since the relaxation times scale[30] as $\ell^{2.2}$, where ℓ is the chain length.

The difficulties in simulating polymer systems stem from these long relaxation times. Long runs are needed in order to ensure adequate equilibration. Equilibrated systems for initializing MD studies can be obtained by the "reptation Monte Carlo" method of Wall and Mandel[31] as modified for continuum polymers by Webman, Kalos and Lebowitz[32]. Starting from some initial configuration one selects one of the ends of the chain at random and places it at the other end with random orientation. The "trial" move is accepted in accordance with the Metropolis criterion. This method has been employed in a number of different polymer investigations[33].

MODEL

The potential for our model consists of two parts: a shifted Lennard-Jones potential[34], U_{LJ} , which operates between all N_p beads in the system and a modified harmonic potential[35], U_H, which links ℓ beads into N chains ($N_p=\ell N$). In terms of the standard reduced units used in MD[25],

$$U_{LJ}(R) = \begin{cases} 4((1/R)^{12} -(1/R)^6 +1/4) & R < 2^{1/6} \\ 0 & R > 2^{1/6} \end{cases} \qquad (1)$$

and

$$U_H(R) = \begin{cases} -0.5kR_0^2 \ln(1-(R/R_0)^2) & 0<R<R_0 \\ 0 & R>R_0 \end{cases} \qquad (2)$$

The Lennard-Jones potential takes account of the excluded volume in a convenient manner in that it is repulsive, short-ranged and continuous. The harmonic potential modifies the simple harmonic potential so that the spring connectors have finite extensibility. We have set R_0 = 1.95 and k = 20. This value of R_0 makes chain crossing unlikely and the value of k makes the time scale of the internal vibrational motion not much shorter than that of the translational motions so that a very small time step, Δt, is not required in the numerical integration routine. The time step was selected to be 0.005. In the simulations, periodic boundaries are used in all three directions. The polymers are initially placed in a box of edge length L set by equating the pre-selected number density (0.3) to N_p/L^3 .

First a serial version of the FORTRAN code was developed. The forces and potentials were tabulated (as is usual practice) so that values are not recomputed from the analytic formulas. Verlet's algorithm[20] was used to move the system in time:

$$\vec{X}_i(t+\Delta t) = 2\,\vec{X}_i(t) - \vec{X}_i(t-\Delta t) + \vec{F}_i(t)(\Delta t)^2 \qquad (3)$$

Here $\vec{X}_i(t)$ and $\vec{F}_i(t)$ are the position and force, respectively, of particle i at time t.

Then the code was modified to conform to the WASHCLOTH simulator; e.g., all shared variables were put into public memory (blank common) and working space was set up for WASHCLOTH (AVAIL). Korn's[36] library routines were used to convert one and two dimensional DO loops to parallel form (i.e.DOALL's or DOWAIT's). If one has the serial DO loop

```
      DO 100 I = ILOW, IHIGH  ISTEP
      DO 100 J = JLOW, JHIGH, JSTEP
      BODY USING I AND J
100   CONTINUE
```

then the parallel form is

```
      ASSIGN 100 TO I
      CALL DOWAIT(KPUB(1,1),I,ILOW,IHIGH,ISTEP,J,JLOW,JHIGH,JSTEP)
      BODY USING I AND J
      CALL ENDDO
100   CONTINUE
```

Here KPUB(1,1) is a public variable which is initialized to zero. It is used to synchronize the PEs executing the DO loops; viz, after a given PE finishes with an I,J set of indices it will obtain another set or wait for the other PEs if there is no more work to be done.

The 'irregular' version of WASHCLOTH, which slows down one or more of the PEs, was used to test the correctness of the parallel code with respect to possible hidden timing coincidences. Also the number of PEs, the number of particles and the number of subboxes were varied. In all cases the serial results for the energies were obtained.

Timing information was extracted by "trapping" a public variable, NTIME, at selected points in the code. This variable is equal to the number of instructions executed---so the running time for any part of the code is found by subtraction of the value of NTIME at the beginning of the section from its value at the end. NTIME is obtained in the simulator by using the PE zero as a master timer; e.g., its "busy-waiting" time as well as its running time are accumulated. Since MD calculations are usually run for thousands of steps, the system initialization times were ignored in the timing analysis; for example, BOX, the routine which sets up the lists of neighboring boxes before starting the time step loop, is called only once in the code and is therefore omitted in the timing analysis. It would be carried out by one PE of a multi-processor system before the beginning of the parallel computation.

Results for one time step are used in this paper even though some of the runs were made for four time steps in order to further check the correctness of the parallel code. The major operations in the loop over time steps include (1) zeroing the force vector (2) adjusting the coordinates for the periodic boundaries (3) finding the subbox a given particle is located in (4) determining the LJ contribution to the force

and energy (5) determining the harmonic contribution to the force and energy and (6) integrating the equations of motion. Operations (1),(2),(3) and (6) are $O(N_p)$, operation (5) is $O(N_p - N)$ which is the number of bonds in the system , and operation (4) is $O(N_b(N_p/N_b)^2)$ where N_b is the number of subboxes. Operation (4) is expected to dominate for large systems.

TABLE I presents timing data for two systems, ℓ=10, N=100 and ℓ=7, N=49, for 1 PE with a variety of values of N_b. The ratios of executed instructions in steps (1), (2), (3) and (6) for the systems are 2.91, 2.93, 2.90 and 2.91 respectively. These compare very well with the expected ratio of 1000/343 = 2.92. The ratio of step (5) for the systems is 3.04 compared to the expected ratio of 900/294 = 3.06. Again there is good agreement. The particle count in the subboxes, $N_c = N_p/N_b$, for the N_p = 1000 system is 15.63, 8.00, 4.63 and 2.92 for N_b = 64, 125, 216 and 343, respectively, whereas it is 5.36, 2.74, 1.59 and 1.00 for the N_p = 343 system for the same number of subboxes. The expected scaling result of $N_b(N_p/N_b)^2$ will not be correct for small particle counts because of the density fluctuations. The particles may not be uniformly distributed in the subboxes and thus the work done by the various PEs can be different. The fluctuations are measured by $\langle N_c^2 \rangle - \langle N_c \rangle^2$. TABLE II contains these quantities for the systems in TABLE I. Except for the case when N_p = 343 and N_b = 343, $\langle N_c^2 \rangle$ becomes closer to $\langle N_c \rangle^2$ as the particle count increases. There is a decrease in the relative fluctuations at higher particle count. The fluctuations are expected to be small for the crystal-like states investigated here.

The effect on efficiency of density fluctuations have not been fully explored in this study since we have not examined fluid-like states.

If we divide the time used for the LJ terms in TABLE I by $N_b\langle N_c^2\rangle$ we obtain for the N_p = 1000 systems 1.127, 1.249, 1.205 and 1.233 for N_b = 64, 125, 216 and 343, respectively, and 1.252, 1.298, 1.691 and 2.860 for the N_p = 343 systems with the same number of boxes. The expected scaling is obeyed by those systems with particle count greater than 1.59.

Runs were then made for 2, 4, 8, 16 and 32 PEs for the various systems. The results are listed, in TABLE III. The three major times are presented, the $O(N_p)$ terms the harmonic term and the Lennard-Jones term as well as, the total time and the speedup and efficiency. Time is measured in 10^3 cycles. Note that the systems with more particles have higher efficiency.

In general[37] if there are L tasks each with executional complexity, T_i, and multiplicity , M_i, and if we assume that the M_i are divided as uniformly as possible among the processors and that each task completes before the next one has begun then the complexity per PE is given by

$$T = \sum_{i}^{L} (F(M_i) \times T_i + K_i) \qquad (4)$$

where

$$F(M) = [M/p] \qquad (5)$$

Here p is the number of PEs, [] is the ceiling function and K_i is the minimal number of instructions that a PE has to execute once it is involved, even if it does no useful work[38].

In our case, the three tasks involve loops over N_p, N and N_b. The respective T_i's will scale as A, B(ℓ-1) and $C\langle N_c^2\rangle$. Using the 2 and 4 PE data of the N_p = 1000, N_b = 343 system one can determine the constants A, B, and C. The results are given by

$$T_{N_p} = 0.932\ [N_p/p\ ] + 3 \qquad (6)$$

$$T_{Harm} = 0.458(\ell-1)\ [N/p] + 1 \qquad (7)$$

$$T_{LJ} = 1.222\langle N_c^2\rangle\ [N_b/p] + 15 \qquad (8)$$

These relationships can be used to predict the results for all the other systems. These predictions are contained in TABLE IV. They agree with the actual timings in TABLE III within 14% for the N_p =1000 systems and 30%, with one exception of 44%, for the N_p = 343 systems except for the lowest particle count of 1.00. There the relative errors are on the order of 55%. Eqs 6-8 can be used to predict the behavior of larger systems such as the ones needed to study the viscosity power law. The speedups and efficiences expected for two such systems are presented in TABLE V. The data demonstrate that the efficiencies of large systems will be high.

A direct computation with the serial code for $\ell = 10$, $N = 100$ systems took about 2.7sec/step on NYU's Cyber computer. A trajectory of 10^3 steps would therefore require 0.75 hours. Since the relaxation times for chains of length ℓ scale as $\ell^{2.2}$, the $\ell = 100$ system would require 158 times as much machine time and the $\ell = 1000$ systems would need 25120 times as much! The speedups obtainable with the parallel code make studies of the dynamic properties of large polymers feasible. That is, a 4096 processor Ultracomputer is projected to take about 8 hours for this calculation.

The speedup of codes can be critically influenced by a number of factors--in particular, the machine architecture and the way in which the algorithm is mapped to that architecture. In the Cm* MIMD machine the processors are connected to memory modules by a distributed switch. "Each computer module includes a local switch, Slocal, which routes processor memory references to memory outside the module via the map bus and accepts references from distant sources to its local memory. Up to 14 computer modules may be connected to a map bus so that they may share the use of a single K map, a processor responsible for mapping addresses and routing data between Slocals...Ignoring the memory performance hierarchy or contention in Cm* can lead to unacceptable degradation."[39]

In a study of parallel solutions to partial differential equations on a Cm* [39] it was found that linear speedups degraded when there was contention for memory containing code. A MD investigation of simple molecular systems[40] (no spring connectors and no box scheme)

demonstrated a "stepped" speedup vs. PE curve; e.g. , there were flat regions in which the speedup did not change as the number of PEs were increased. The peaks, however, indicated linear speedup. These results were attributed to the unequal distribution of work among PEs in the flat regions of the curve.

The results of these two studies are a consequence of the increased effect of the hierarchical memory as more PEs are brought to bear on a problem of fixed size. The NYU Ultracomputer design in which the switches permit simultaneous access to memory locations will remove this difficulty. Moreover, the inclusion of a cache will improve the performance significantly. An analysis[41] of the memory references generated by the code has shown that at least 95% of those carried out in the parallel loops were to instructions or to private data. These are all easily cacheable. However, even without the hierarchical memory effects the overhead required to introduce more PEs for a fixed problem size, on any machine, can cause performance degradation. This is accounted for in our study by the K_1 term in eq.4 but the simulation results show this term to be small.

On lattice connected machines, the calculation of the intrachain forces would be slow when the chains are spread across the simulation box. Thus, the type of architecture planned for the NYU Ultracomputer should make this machine an excellent vehicle for polymer simulations on the very large scale required to answer serious questions related to dynamic scaling phenomena.

Acknowledgements

I would like to thank Malvin H. Kalos for suggesting this investigation and for his illuminating comments on parallel programming. I would also like to thank David Korn , Boris Lubachevsky and Kevin McAuliffe for helpful discussions.

References

1. I. E. Sutherland and C. A. Mead, Sci. Am., Sept., 210(1977)

2. N. S. Ostlund, NRCC Report, LBL-10409 UC-32, Jan. (1980) See also N. S. Ostlund, Int. J. Quant. Chem., Quant. Chem. Sym., 13 15(1979)

3. M. J. Flynn, IEEE Trans. Comp., Sept., 948 (1972)

4. W. J. Karplus, Simulation, 29 143 (1977)

5. R. W. Hockney and C. R. Jesshope, Parallel Computers, (Adam Hilger Ltd, Bristol, 1981)

6. H. S. Stone, Introduction to Computer Architecture, (Science Research Associates, Chicago, 1975)

7. J. Schwartz, JACM, 13 25(1966)

8. J. Schwartz, TOPLAS, 2 484(1980)

9. A. Gottlieb, Ultracomputer Notes 12 and 21, Courant Institute, NYU 1980-81.

10. The code was actually implemented with a replace-add instruction which returns the sum, V + e, and replaces the contents of storage location V by that sum. If replace-add is used instead of fetch-and-add in the example given V will still contain 18 even though now S1=12 and S2=18 or S1=18 and S2=16.

11. A. Gottlieb and J. Schwartz, Computer, Jan., 27 (1982)

12. A. Gottlieb, R. Grishman, C. P. Kruskal, K. P. McAuliffe, L. Rudolph and M. Snir, IEEE Trans. Comp., C32 175 (1983)

13. M. H. Kalos, Ultracomputer Note 48, Courant Institute, NYU, 1983

14. M. H. Kalos, Ultracomputer Note 30, Courant Institute, NYU, 1981

15. P. J. Flory, Principles of Polymer Chemistry, (Cornell University Press, New York, 1953)

16. F. T. Wall, S. Windwer and P. J. Gans, Methods of Computational Physics, (Academic Press, New York, 1963) Vol. I and S. Windwer, Markov Chains and Monte Carlo Calculations in Polymer Science, ed. G. G. Lowry (Dekker, New York, 1970)

17. See for example S. D. Stellman and P. J. Gans, Macromolecules, 5 516 (1972); R. Grishman, J. Chem. Phys., 58 220 (1973); and F. L. McCrackin, J. Mazur and C. M. Guttman, Macromolecules, 6 859 (1973)

18. N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller and E. Teller, J. Chem. Phys., 21 1087 (1953)

19. W. W. Wood and J. J. Erpenbeck, Ann. Rev. Phys. Chem., 27 319 (1976)

20. L. Verlet, Phys. Rev., 159 98 (1967)

21. B. Quentrec and C. Brot, J. Comput. Phys., 13 430 (1973)

22. M. H. Kalos, G. Leshen and B. D. Lubachevsky, Ultracomputer Note 26, Courant Institute, NYU 1981

23. J. P. Rychaert and A. Bellemans, Chem. Phys. Lett., 30 123 (1975)

24. T. A. Weber, J. Chem. Phys., 69 2347 (1978)

25. M. Bishop, M. H. Kalos and H. L. Frisch, J. Chem. Phys., 70 1299 (1979)

26. D. C. Rapaport, J. Chem. Phys., 71 3299 (1979)

27. D. W. Rebertus, B. J. Berne and D. Chandler, J. Chem. Phys., 70 3395 (1979)

28. W. Bruns and R. Bansal, J. Chem. Phys., 74 2064 (1981)

29. R. S. Porter and J. F. Johnson, Chem. Rev., 66 1 (1966)

30. D. Ceperley, M. H. Kalos and J. L. Lebowitz, Phys. Rev. Lett., 41 313 (1978)

31. F. T. Wall and F. Mandel, J. Chem. Phys., 63 4592 (1975)

32. I. Webman, J. L. Lebowitz and M. H. Kalos, J. Physique, 41 579 (1980)

33. M. Bishop, D. Ceperley, H. L. Frisch and M. H. Kalos, J. Chem. Phys., 72 3228 (1980) and M. Bishop, D. Ceperley, H. L. Frisch and M. H. Kalos, J. Chem. Phys., 76 1557 (1982)

34. J. D. Weeks, D. Chandler and J. C. Anderson, J. Chem. Phys., 54 5237 (1971)

35. R. C. Armstrong, J. Chem. Phys., 60 724 (1974)

36. D. Korn, Ultracomputer Note 23, Courant Institute, NYU (1981)

37. B. D. Lubachevsky, Ultracomputer Note 42, Courant Institute, NYU(1982)

38. D. Korn, Ultracomputer Note 24, Courant Institute, NYU (1981)

39. K. Jones and P. Schwarz, Comp. Surveys, 12 121 (1980)

40. N. S. Ostlund, P. G. Hibbard and R. A. Whiteside, Parallel Computations, ed. G. Rodrigue (Academic Press, New York, 1982) p.315

41. K. McAuliffe, private communication

TABLE CAPTIONS

TABLE I: Timing results for 1 PE for different chain lengths, ℓ and different number of chains, N, as a function of the number of subboxes, N_b. Units are in 10^3 instructions.

TABLE II : The measured values of $\langle N_c^2 \rangle$ and $\langle N_c \rangle^2$ for all the systems

TABLE III : The simulation results for different numbers of PEs for all the systems T_{N_p}, T_{Harm} , T_{LJ} and T_{Total} are the times for N_p terms, harmonic, LJ and the total, respectively. Time is in units of 10^3 instructions. The speedup and efficiency are defined in the text.

TABLE IV : The predicted efficiencies for the different systems

TABLE V : The predicted total time, T, Speedup and efficiences of larger systems.

TABLE I

$\ell = 10$, N = 100

N_b	64	125	216	343
1)zero force	186	186	186	186
2)adjust coordinates	234	234	234	234
3)locate subbox	235	235	235	235
4)LJ terms	17937	9994	6441	4510
5)harmonic terms	414	414	414	414
6)integration	282	282	282	282

$\ell = 7$, N = 49

1)zero force	64	64	64	64
2)adjust coordinates	80	80	80	80
3)locate subbox	81	81	81	81
4)LJ terms	2510	1509	1155	987
5)harmonic terms	136	136	136	136
6)integration	97	97	97	97

TABLE II

$N_p = 1000$

$N_b =$	64	125	216	343
$\langle N_c^2 \rangle$	248.656	64.000	24.750	10.665
$\langle N_c \rangle^2$	244.141	64.000	21.437	8.497

$N_p = 343$

$N_b =$	64	125	216	343
$\langle N_c^2 \rangle$	31.328	9.304	3.162	1.006
$\langle N_c \rangle^2$	28.719	7.530	2.522	1.000

TABLE III

ℓ = 10, N = 100

Number of PEs	T_{N_p}	T_{Harm}	T_{LJ}	T_{Total}	Speedup	Efficiency
	N_b = 64					
1	937	414	17937	19288	1.00	1.00
2	469	207	9035	9711	1.99	1.00
4	236	104	4704	5044	3.82	0.96
8	120	54	2437	2611	7.39	0.92
16	60	29	1369	1458	13.23	0.83
	N_b = 125					
1	937	414	9994	11345	1.00	1.00
2	469	207	5038	5714	1.99	1.00
4	236	104	2562	2902	3.91	0.98
8	120	54	1314	1488	7.62	0.95
16	60	29	705	794	14.29	0.89
32	31	17	381	429	26.45	0.83
	N_b = 216					
1	937	414	6441	7792	1.00	1.00
2	469	207	3241	3917	1.99	1.00
4	236	104	1636	1976	3.94	0.99
32	31	17	250	298	26.15	0.82

N_b =343

1	937	414	4510	5861	1.00	1.00
2	469	207	2257	2933	2.00	1.00
4	236	104	1136	1476	3.97	0.99
8	120	54	574	748	7.84	0.98
16	60	29	294	383	15.30	0.96
32	31	17	155	203	28.87	0.90

$\ell = 7$ N = 49 $N_b = 64$

Number of PEs	T_{N_p}	T_{Harm}	T_{LJ}	T_{Total}	Speedup	Efficiency
1	322	136	2510	2968	1.00	1.00
2	162	70	1279	1511	1.96	0.98
4	80	36	685	801	3.71	0.93
8	40	20	391	451	6.58	0.82
16	20	11	242	273	10.87	0.68
32	11	6	165	182	16.31	0.51

$N_b = 125$

1	322	136	1509	1967	1.00	1.00
2	162	70	759	991	1.98	0.99
4	80	36	382	498	3.95	0.99
8	40	20	200	260	7.57	0.95
16	20	11	112	143	13.76	0.86
32	11	6	66	83	23.70	0.74

$N_b = 216$

1	322	136	1155	1613	1.00	1.00
2	162	70	578	810	1.99	1.00
4	80	36	293	409	3.94	0.99
8	40	20	149	209	7.72	0.97
16	20	11	80	111	14.53	0.91
32	11	6	43	60	26.88	0.84

$N_b = 343$

1	322	136	987	1445	1.00	1.00
2	162	70	495	727	1.99	1.00
4	80	36	248	364	3.97	0.99
8	40	20	125	185	7.81	0.98
16	20	11	64	95	15.21	0.95
32	11	6	33	50	28.90	0.90

TABLE IV

N_p = 1000

Number of PEs	N_b = 64 T_{N_p}	T_{Harm}	T_{LJ}	N_b = 125 T_{N_p}	T_{Harm}	T_{LJ}
1	935	413	19462	935	413	9791
2	469	207	9738	469	207	4942
4	236	104	4877	236	104	2518
8	120	55	2446	120	55	1266
16	62	30	1230	62	30	641
32	33	17	623	33	17	328

	N_b = 216			N_b = 343		
1	935	413	6548	935	413	4485
2	469	207	3281	469	207	2257
4	236	104	1648	236	104	1136
8	120	55	832	120	55	575
16	62	30	438	62	30	302
32	33	17	227	33	17	158

N_p = 343

N_b = 64				N_b = 125		
Number of PEs	T_{N_p}	T_{Harm}	T_{LJ}	T_{n_p}	T_{Harm}	T_{LJ}
1	323	136	2465	323	136	1436
2	163	70	1240	163	70	731
4	83	37	628	83	37	379
8	43	20	321	43	20	197
16	24	12	168	24	12	106
32	13	6	92	13	6	60

N_b =216				N_b = 343		
1	323	136	850	323	136	437
2	163	70	432	163	70	226
4	83	37	224	83	37	121
8	43	20	119	43	20	68
16	24	12	69	24	12	42
32	13	6	42	13	6	29

TABLE V

$\ell = 100$, N = 100, $N_b = 343$, $N_p/N_b = 29.15$

Number of PEs	T_{N_p}	T_{Harm}	T_{LJ}	T_{Total}	Speedup	Efficiency
1	9323	4535	356173	370031	1.00	1.00
32	295	182	11437	11914	31.06	0.97
512	22	46	1053	1121	330.09	0.64
4096	6	46	1053	1105	334.87	0.08

$\ell = 1000$, N= 100, $N_b = 3375$, $N_p/N_b = 29.63$

Number of PEs	T_{N_p}	T_{Harm}	T_{LJ}	T_{Total}	Speedup	Efficiency
1	93203	45755	3620846	3759804	1.00	1.00
32	2916	1831	113736	118483	31.73	0.99
512	186	459	7525	8170	460.20	0.90
4096	26	459	1088	1573	2390.20	0.58